How to create Brilliant CF Conventions

John Caron
UCAR/Unidata
Feb 25, 2015


CF 2.0

Standards

Cant live with them, cant live without them

Confessions of a Software Library Writer

Scattered thoughts about Standards and Data Models

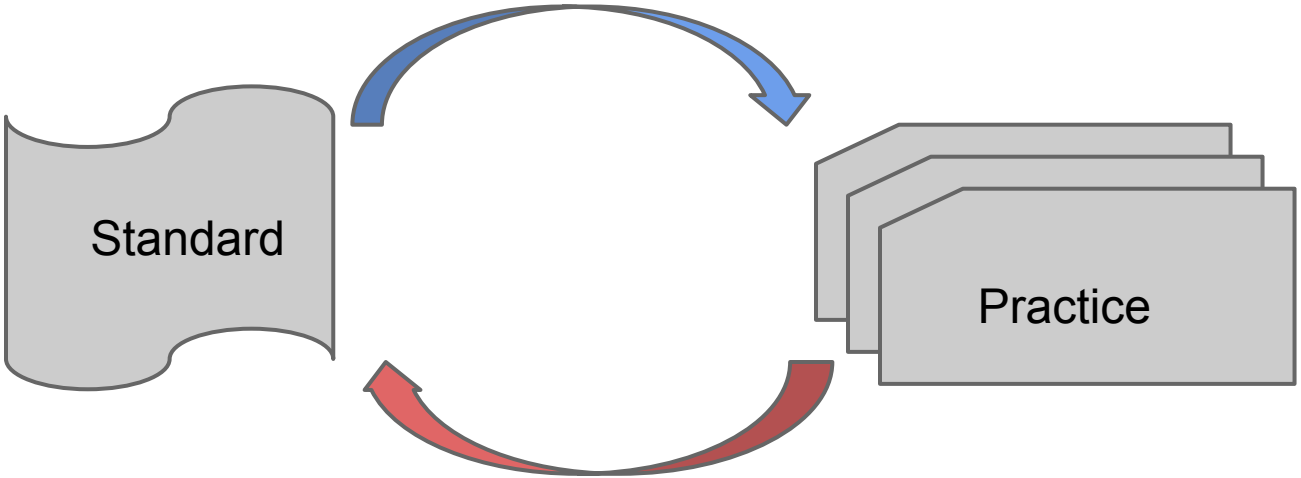
For earth-science data

As told by a writer of software libraries and tools

Overview

- Hand-waving about standards (OGC, WMO, CF)
- Data Models
- ~~What is a Dimension?~~
- Grib Collections
- Big Data
- ~~Postmodernism~~
- ~~Java vs Python~~

Which Comes First?



- Need a standard before you can implement it.
- Need a “best” practice before you can standardize it.

Creating standards: Necessary tensions

Abstraction: general vs specific

Control: constraint vs anarchy

Idealism: standard (idea) vs implementation (code)

Insularity: internal vs external

Revolution: backwards compatibility vs innovation

Reproduction: incremental vs syncretic

Creating standards: Musts

1. Must have an iterative process
2. Must separate semantics from encoding
3. Must enable a community of practice

   Clarify, refine, get advice, document, create examples

4. Must implement in software

   Ideally 2+ independent implementation

   Ideally a reference implementation

   Must test interoperability

5. Must have feedback loops

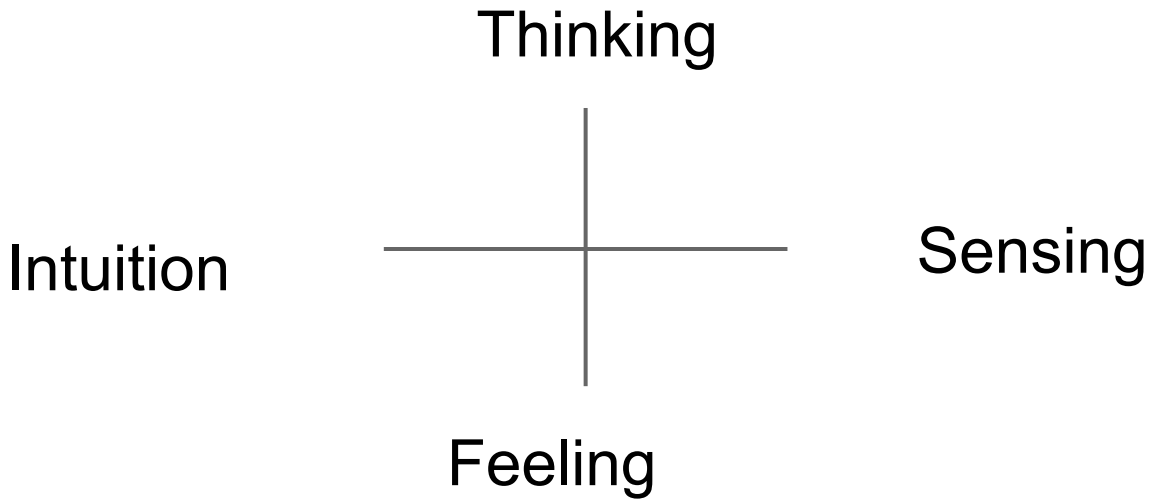
Standards

- OGC - WCS, WMS, O&M, SOS, ...
  - abstract, formal, prescriptive, conformance
- WMO - GRIB, BUFR
  - pragmatists, controlled vocabularies
  - WMO only, closed process
- CF
  - pragmatists, narrow scope
  - netCDF-centric
  - ad-hoc, bespoke

Creating Standards: Process

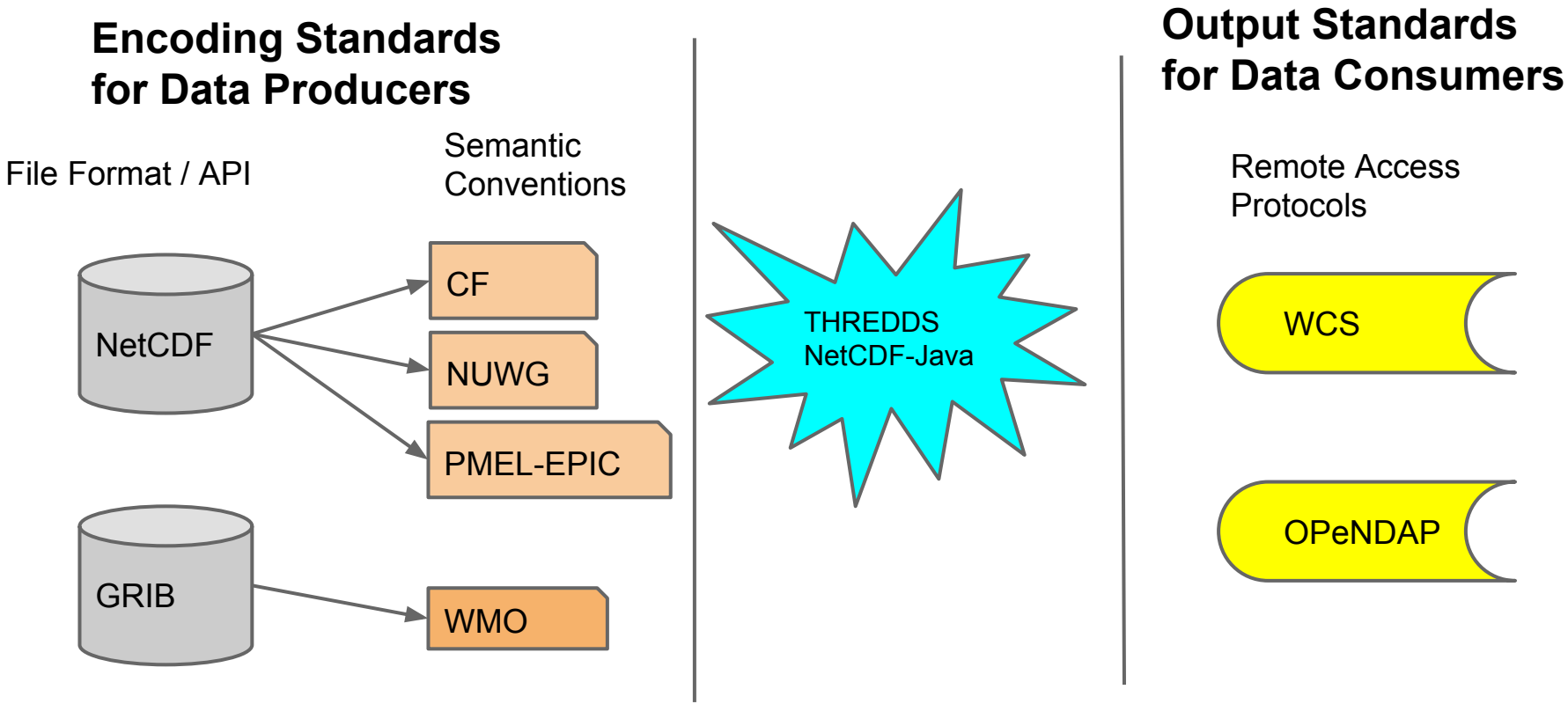
- The standards process is a group process
  - small vs large number of people; email vs face-to-face
  - requires management as well as technical expertise
  - requires emotional intelligence
- The standards document reflects human understanding
  - describes the Convention, it is not the Convention
  - examples, FAQ, software, conformance, standard tests
- The Convention reflects software understanding
- Conventions need versioning
- You will only work on problems you need to solve

Carl Jung's "ways of knowing"

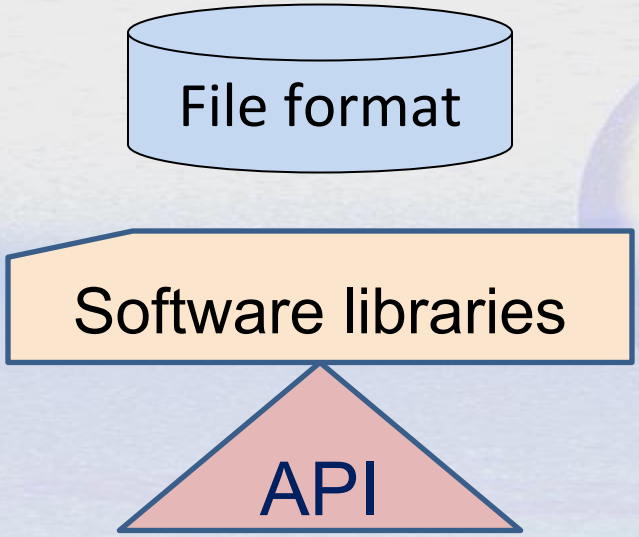


"Under **sensation** I include all perceptions by means of the sense organs; by **thinking**, I mean the function of intellectual cognition and the forming of logical conclusions; **feeling** is a function of subjective evaluation; **intuition** I take as perception by way of the unconscious, or perception of unconscious events"

Where does THREDDS fit in?



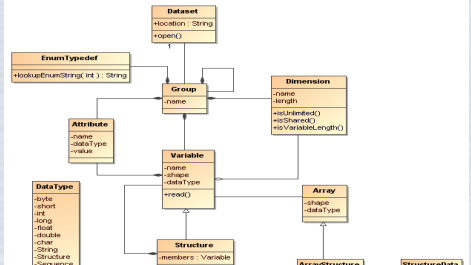
What is NetCDF?



Store data model objects
Persistence layer
NetCDF-3, netCDF-4

Implements the API
C, Java, Python

An ***API*** is the interface to the Data Model for a specific programming language

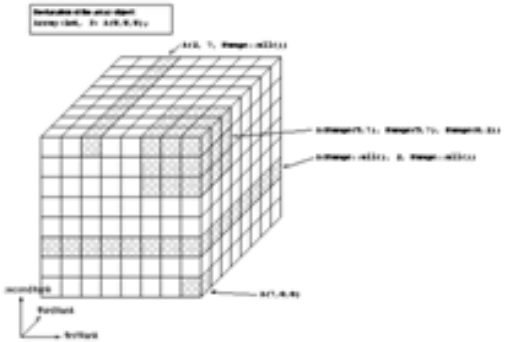


An **Abstract Data Model** describes data objects and what methods you can use on them

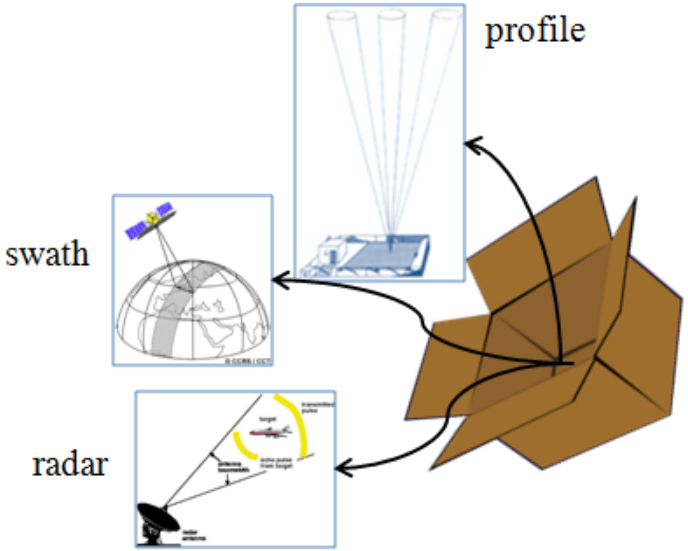
Whats in a file?

OS File:
Bag of Bytes

NetCDF/HDF File:
Multidimensional Arrays

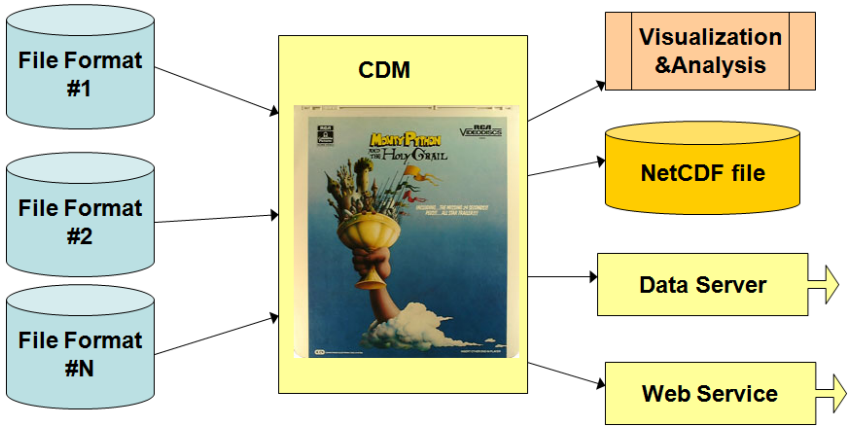


Collection of Objects
Feature Types:



What is our plan for World Domination?

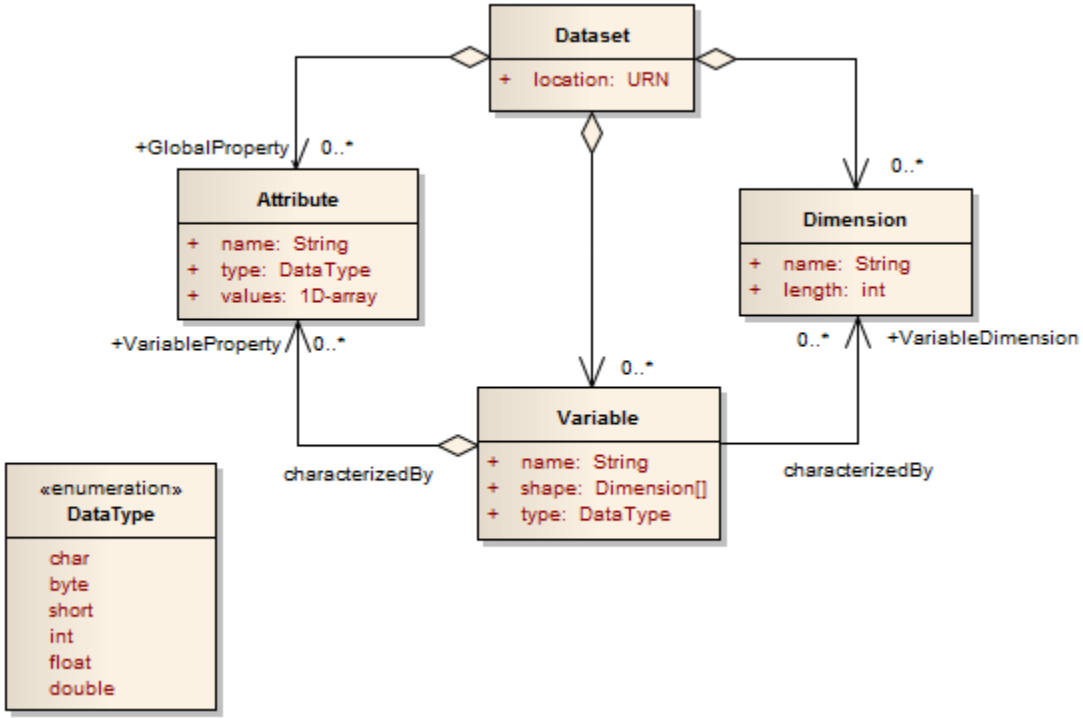
- Independent implementation of NetCDF in Java - portability
- But meteorology doesn't store data in netCDF (GRIB, BUFR)
- For app, its the API that matters, not the file format
- Make everything look like netCDF, access through common API
- N*M ➔ N+M
- Its the data model, stupid
- Common Data Model (CDM)



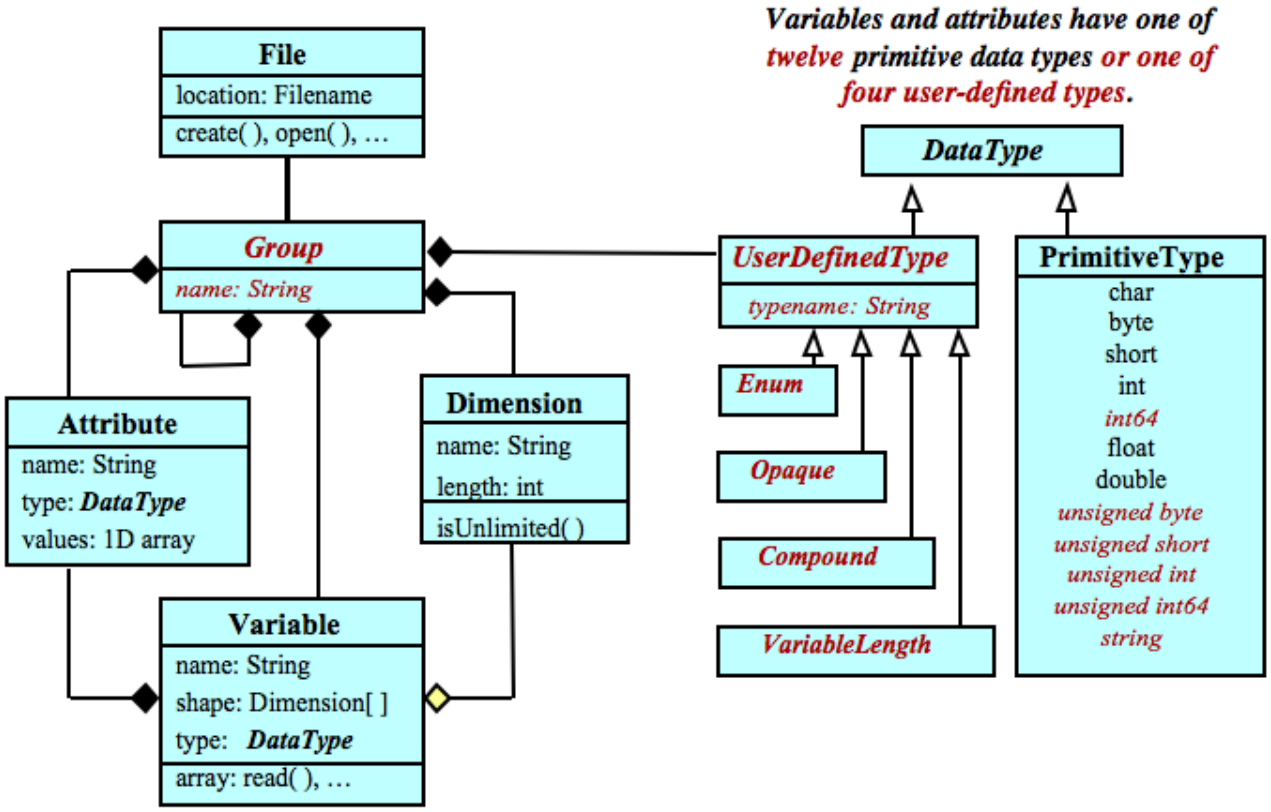
What is a Data Model?

- Language for talking about your problem independent of the details of encoding and language
- A simplification of your problem to emphasize its salient features
- If you are a software engineer, you will use UML (Unified Modeling Language)

NetCDF Classic model

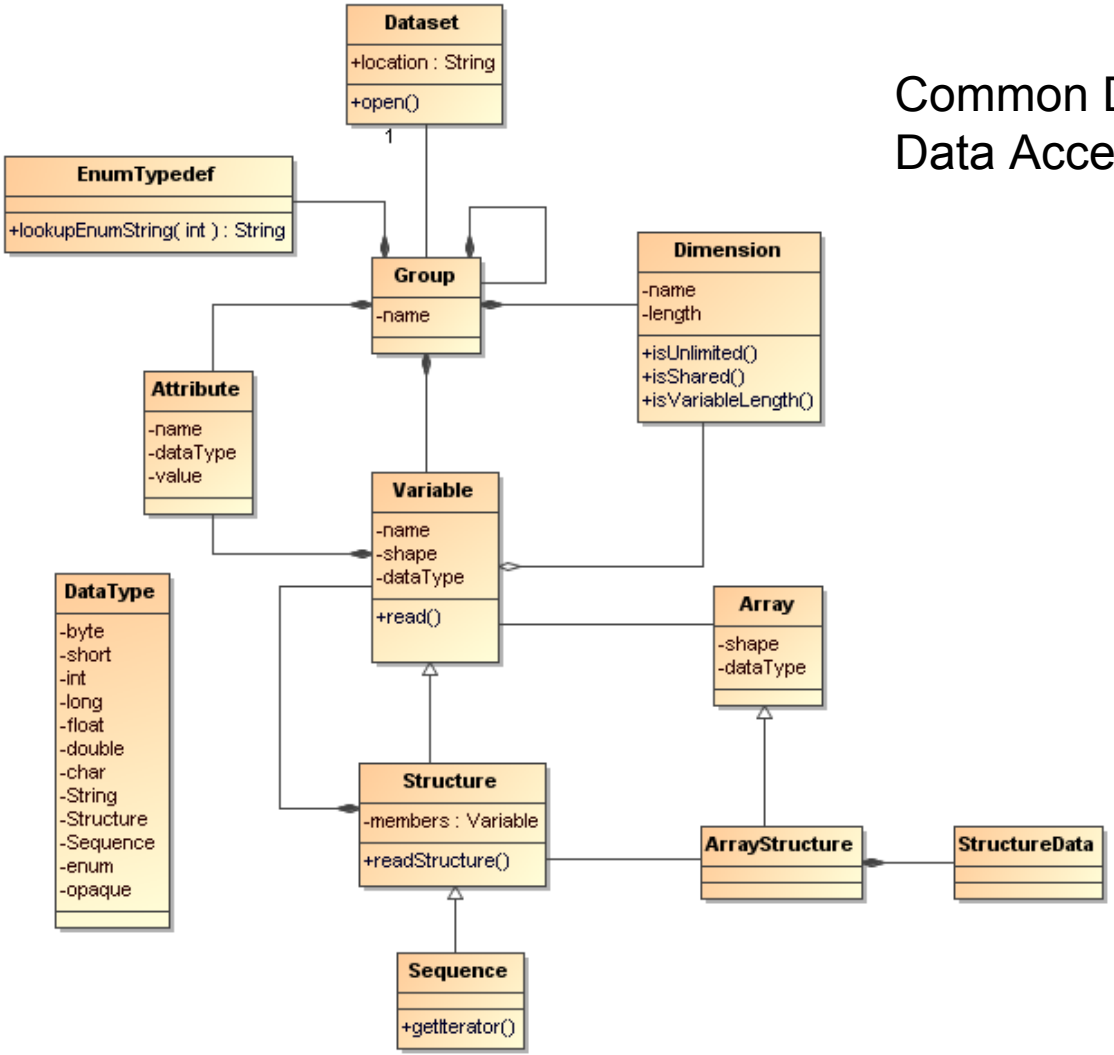


NetCDF-4 Data Model

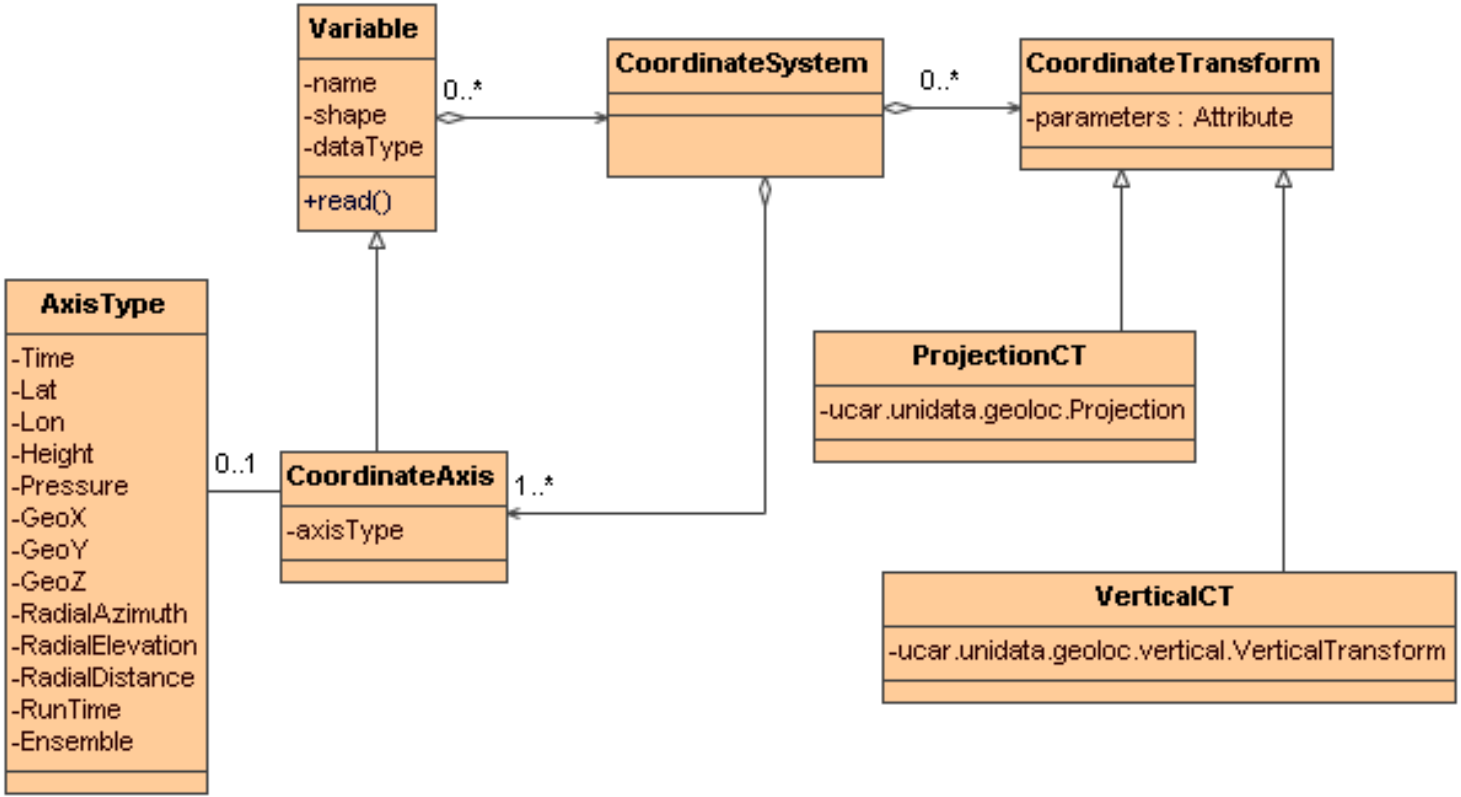


A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

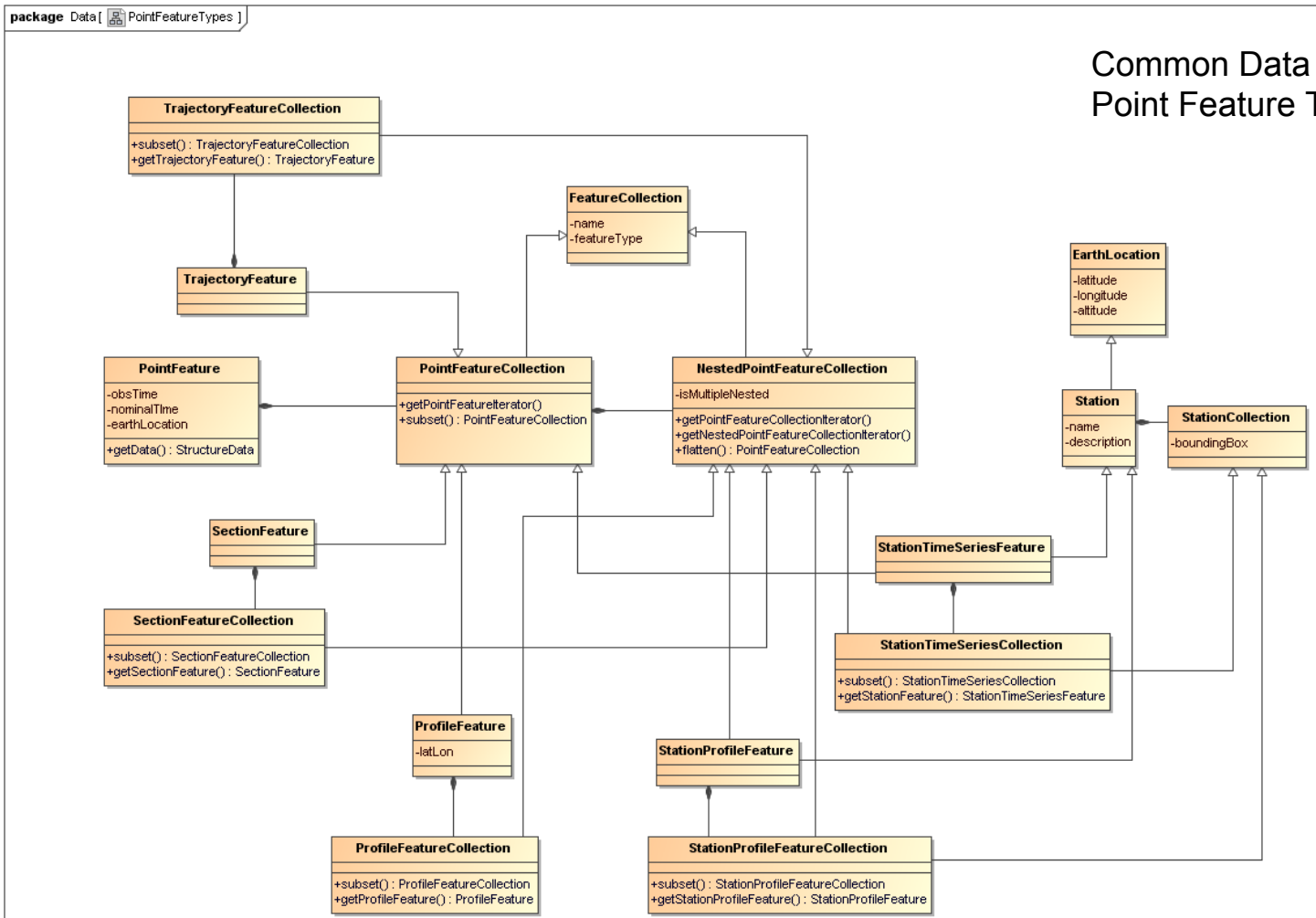
Common Data Model
Data Access Layer

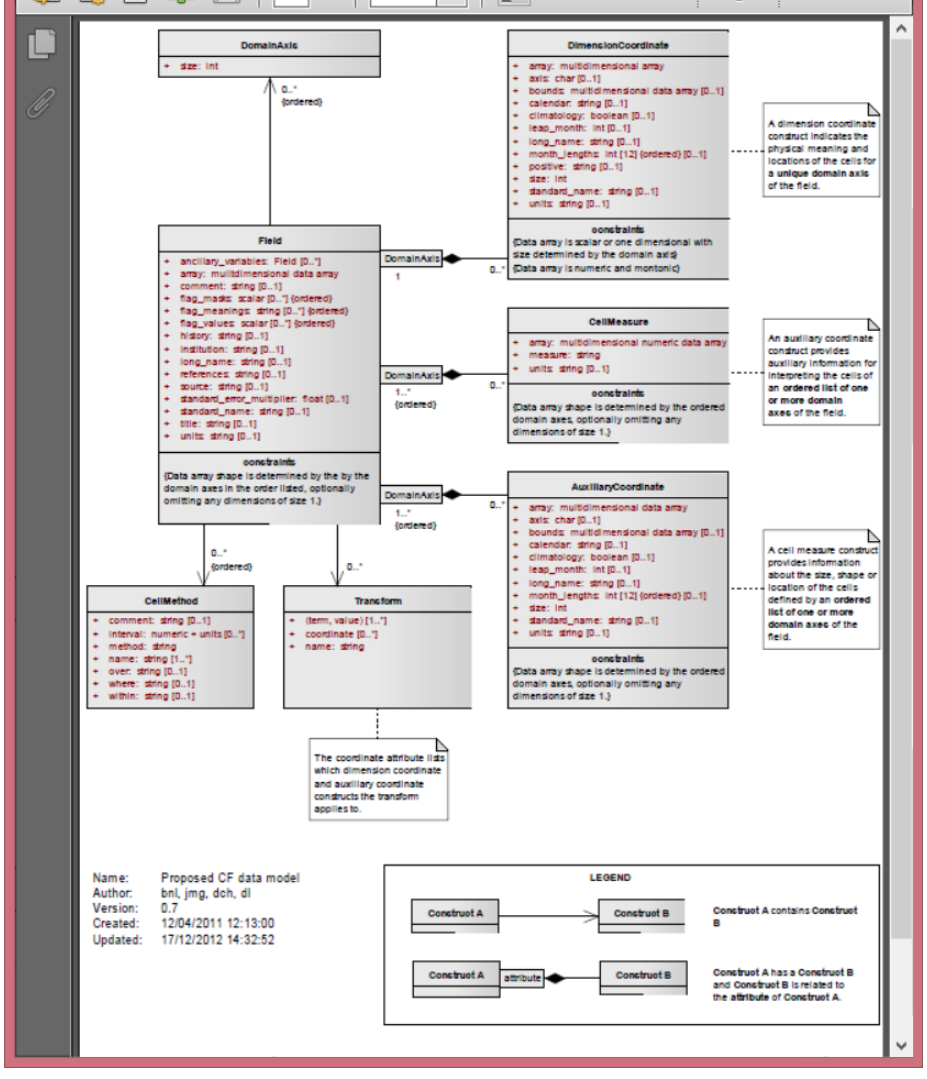


Common Data Model
Coordinate Systems Layer

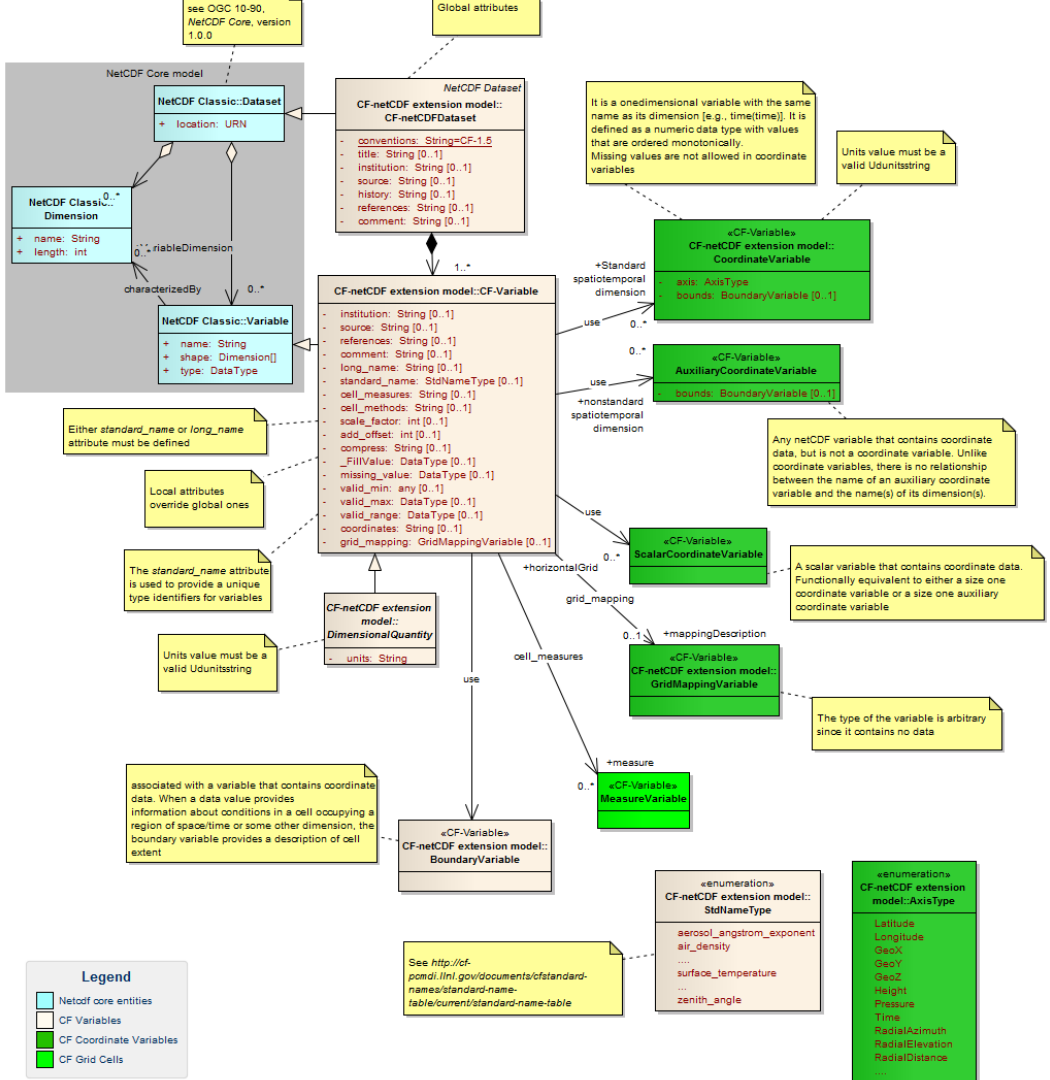


Common Data Model
Point Feature Types





Proposed CF Data Model
17/12/2012

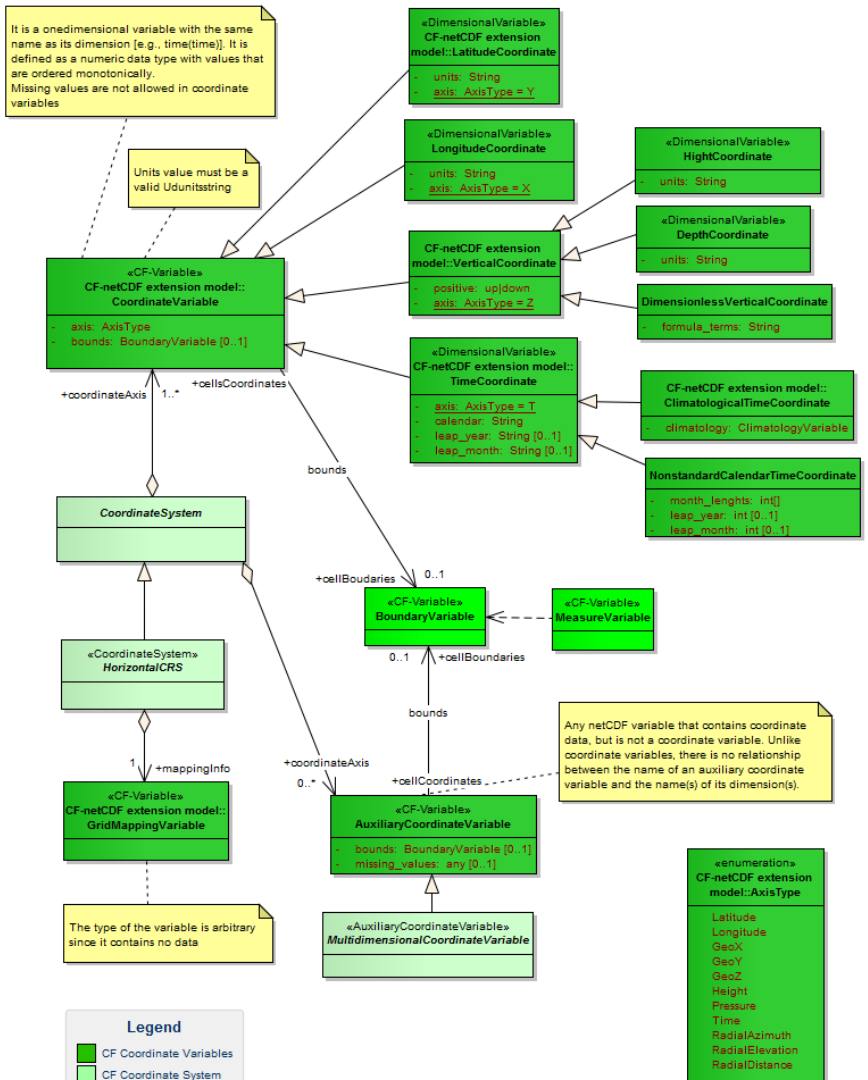


OGC Standard
CF-netCDF Data Model
CF Variable and Standard Attributes
Domenico and Nativi
2012-Aug 12

CF Coordinate Variables, Coordinate Types, Coordinate Systems, and Grid Cells

Domenico and Nativi

2012-Aug 12

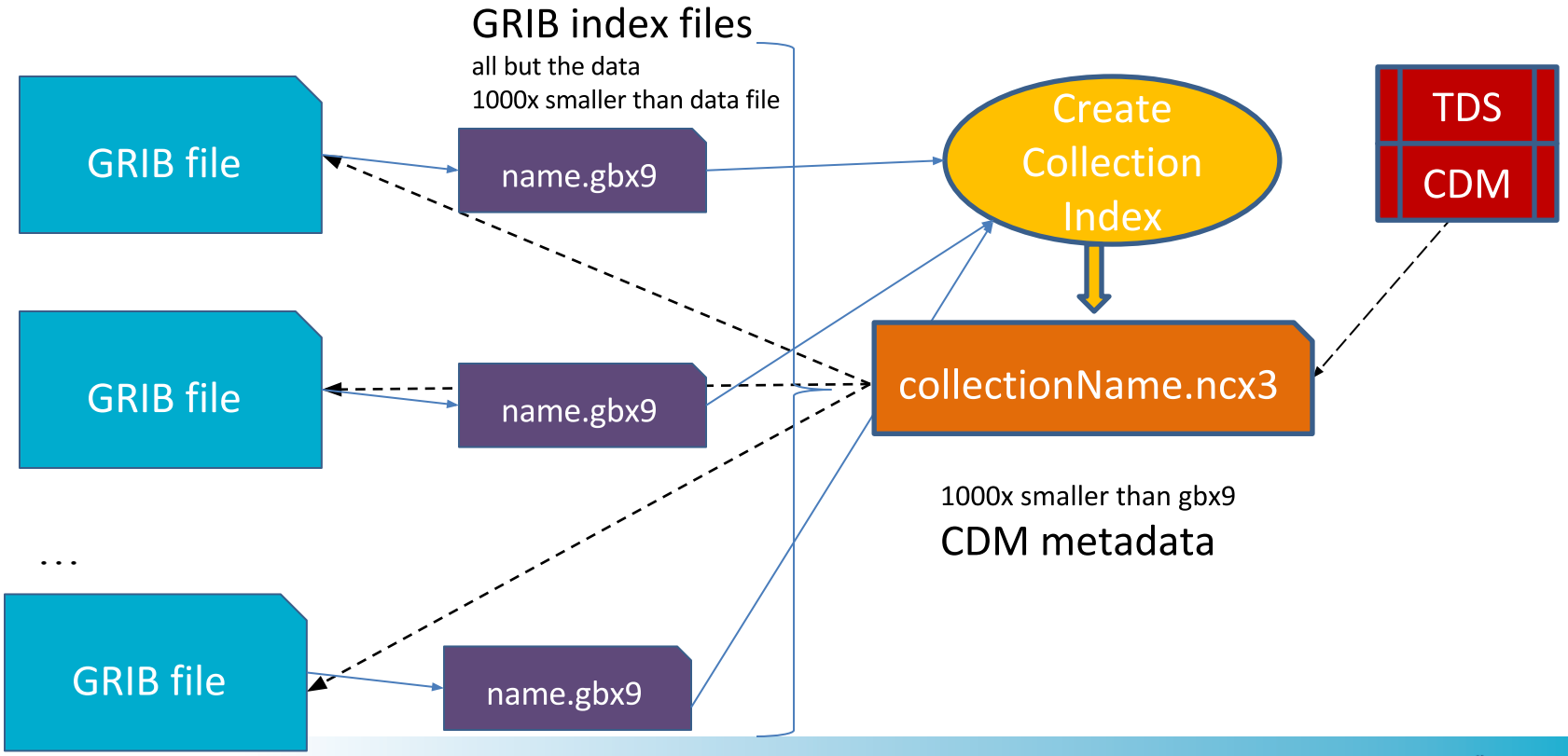


OGC CF-NetCDF Requirements (67)

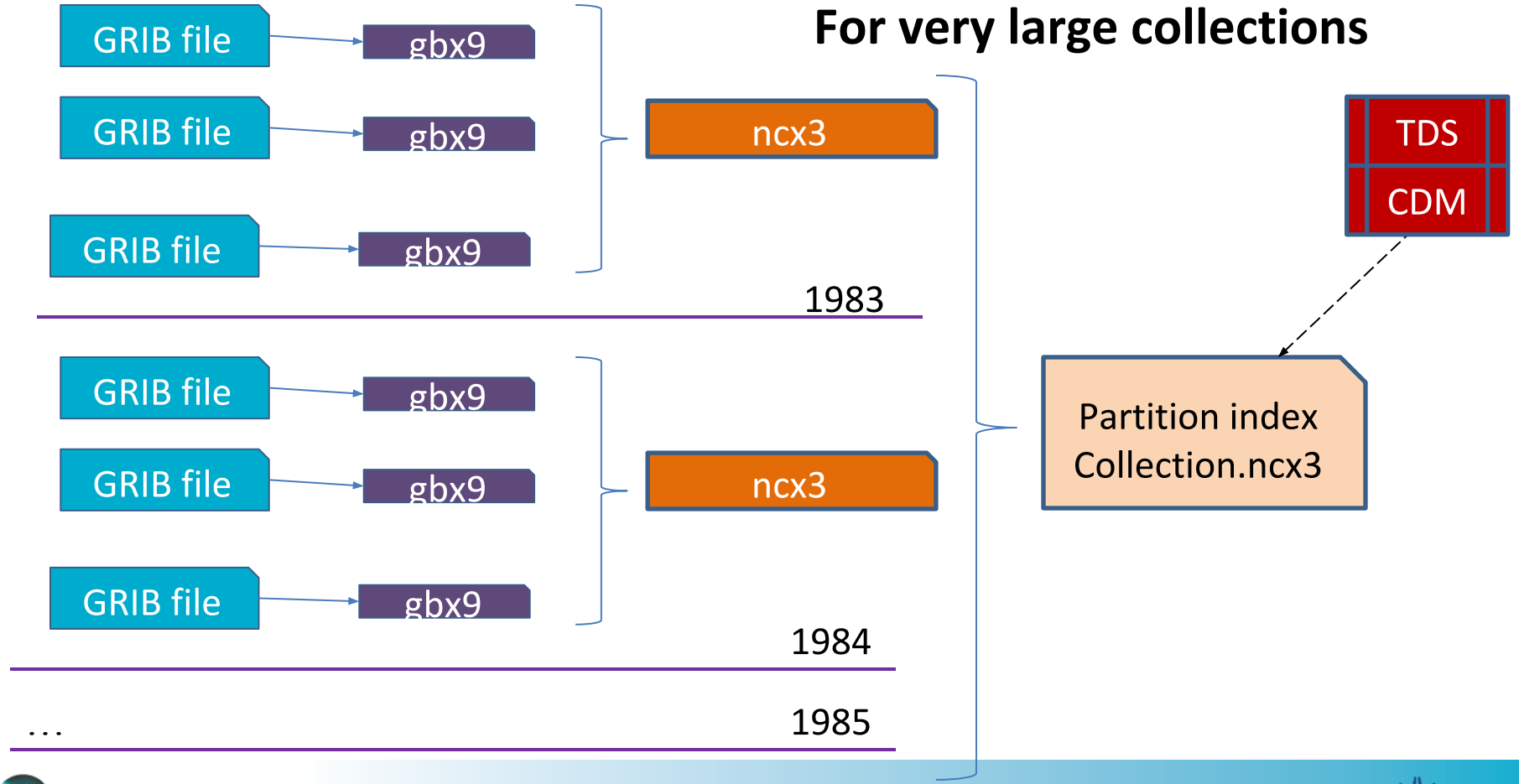
- *Any data instantiating a concrete CF-netCDF dataset **shall** conform with the UML diagrams in Figure 3 and Figure 4*
- *Any CF-netCDF Dataset that uses the CF convention **shall** define the global attribute Conventions to the string value "CF-1.6".*
- *For a given spatial-temporal CF-netCDF Variable, its spatial-temporal Dimensions order **shall** appear in the relative order T, then Z, then Y, then X. In addition, any other dimension shall be placed to the left of the spatiotemporal dimensions.*
- *Any Time Coordinate **shall** define the calendar attribute…*
- *...*

GRIB Collections

GRIB collection indexing



GRIB time partitioning (nested heirarchies)
For very large collections



GRIB Feature Collections

NCAR Research Data Archives (RDA)

	#files	#records	#vars	#runtimes	#Gbytes
NCEP FNL Global Tropospheric Analyses	22093	6473541	182	22093	1700
NOAA CIRES 20th Century Global Reanalysis Version 2	40468	152460254	113	209164	13600
NCEP Global Ocean Data Assimilation System (GODAS)	415	85905	12	415	117
Subsurface Temperature/Salinity Analyses (1945-present)	402	231552	4	816	16
ECMWF Reanalysis Interim Project	51860	363020	7	51860	33930

...

makes these available as multidimensional arrays

```
float data(reftime, ens, time, z, y, x)
```

creates logical views (2D time, best, analysis-only, …)

GRIB -> NetCDF Issues

1 External Table Nightmare
- can be solved if you are the data provider

2 Data Model Differences
- record oriented GRIB vs multidimensional NetCDF
- What is a Variable (and what is its name)?
- can be solved if you are the data provider
- generally impossible due to data model mismatch

3 Missing records in multidimensional array

4 File Sizes / Compression
- adding compression options to compete with GRIB

Compression

- On NCEP Model GRIB files “limited precision” floats, bzip2/grib size vary between .4 and 1.7, average 1.2
- Bzip2 looks like a good candidate to add as a standard compression option in netCDF-4 in addition to deflate
- Idea is to offer options that tradeoff file size and un/compress times
- We are considering a “lossy compression” option in netCDF-4 using bit shaving and/or scale/offset
- Other compression options still to explore: fpzip, zfp from Peter Lindstrom (llnl), libaec from Luis Kornbluth (dkrz)

Big Data

Big Data in the Cloud

How to deal with the cost of providing access to large amounts of data?

Put data in the cloud and let the cloud provider charge

- customers to spin up virtual machine(s) to process
- per byte to download

Allow third parties to add value, provide services, etc.

Assumes data is unrestricted

Bigger Data

- CMIP5 at IPSL Climate Modelling Centre
  - 300K netCDF files, 300 Tb.
- Sequential read of 300 Tb @ 100Mb/sec
  - 3 x 10^6 sec = 833 hours = 35 days
- How to get that down to 8 hours ?
  - Divide into 100 jobs, run in parallel
- What could you do if it was possible to read through CMIP5 overnight?

Required: Parallelizable High Level Language

- Scientific Data Management in the Coming Decade, Jim Gray (2005)
  - Now: File-at-a-time processing in Fortran
  - Need: Set-at-a-time processing in HLQL
    - Declarative language like SQL (vs. procedural):
    - Define dataset subset to work against
    - Define computation
    - Let the system figure out how to do it

Required: Send User programs to server

- Need a query / computation language
  - easily parallelized
  - scientists can/will use
  - Powerful enough to accomplish hard tasks
- What language?
  - Not Fortran, C, Java (too low level)
  - Some subset / dialect of Python ? (Probably not)

Some Existing Candidates

- SciDB ArrayQL
- OGC Web Coverage Processing Service (WCPS)
- Google Earth Engine
- IRIS Data Language
- Ferret
- R

“Coordinate Space” Data Access:

http://motherlode.ucar.edu:8080/thredds/ncss/grid/
NAM_CONUS_80km_20081028_1200.grib1?
var=Precipitable_water&
time=2008-10-28T12:00:00Z&
north=40&south=22&west=-110&east=-80

Fake SQL:

SELECT Precipitable_water
FROM NAM_CONUS_80km_20081028_1200.grib1
WHERE time=2008-10-28T12:00:00Z
AND space=[north=40,south=22,west=-110,east=-80]

More Elaborate

DATASET cfsr

FROM CFSR-HPR-TS9

WHERE *month=April* **AND** *year >=* 1990

AND *space*=[north=40,south=22,west=-110,east=-80]

AS *Grid*

SELECT precip=Precipitable_water, rh=Reletive_Humidity,
T=Temperature

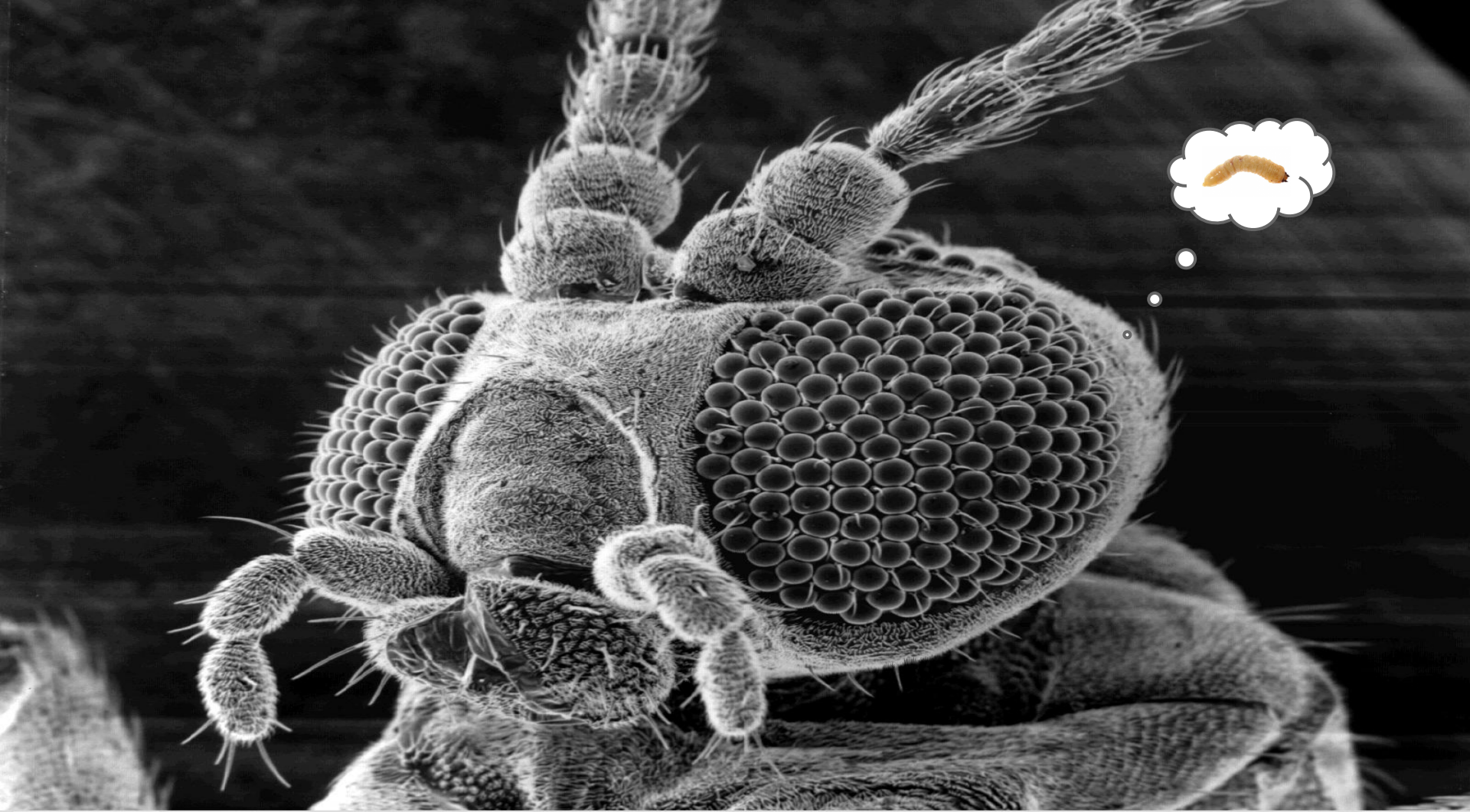
FROM cfsr

CALC *DailyAvg(Correlation(* precip*,* rh*) / Avg(*T*))*

RETURN AS *Grid*

Summary: Big Data Post-Processing

- Need parallel I/O System
  - Shared nothing, commodity disks, replicated data
- Need parallel processing system
  - like Hadoop
- Need to send computation to the server
- Need a parallelizable query / computation language
  - Declarative
  - Must be expressive and powerful
  - Probably a new “domain specific” language


50 µm
26Cecidomyiidae200
1/6/0 REMF

Dimensions

What is a netCDF Dimension?

1. Defines shape of a multidimensional array
2. **Ordering** defines the layout on disk, and thus the cost of accessing array sections
   - NetCDF-4 chunking gives lots more user control
3. **Sharing** enables coordinate systems
   - use netCDF-4 instead of HDF-5
4. Coordinate Systems are needed for georeferencing

But there isnt a clear understanding in CF Community of how to use Dimensions, ie of Coordinate Systems

Example of station data
Whats wrong?

```
dimensions:
   lat = 193;
   lon = 193;
   time = 24;
variables:
   float data(lat, lon, time);
      :coordinates = “lat lon time”;
  float lat(lat);
  float lon(lon);
  int time(time);
```

Only “works” when lat=1, lon=1

Correct

Orthogonal multidimensional array representation of time series

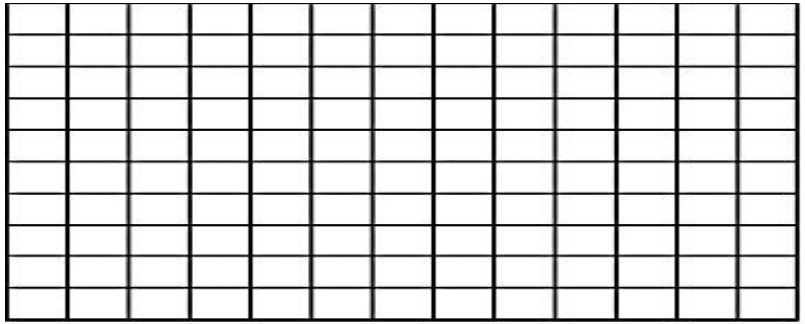
```
dimensions:
  station = 193;
  time = 24;
variables:
  float data(station, time);
    :coordinates = "lat lon time";
 float lat(station);
 float lon(station);
 int time(time);
```

Correct

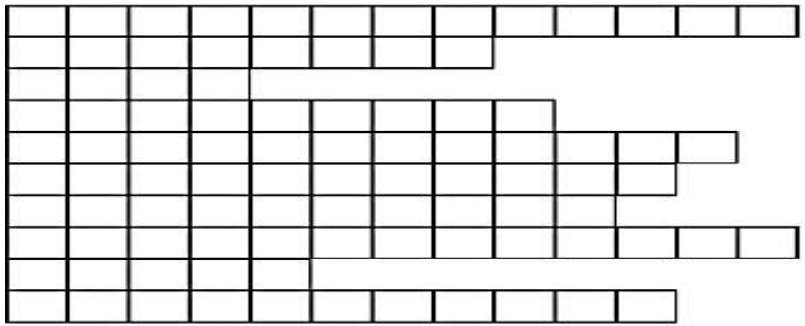
Incomplete multidimensional array representation of time series

```
dimensions:
  station = 193;
  time = 24;
variables:
  float data(station, time);
    :coordinates = "lat lon time";
 float lat(station);
 float lon(station);
 int time(station, time);
```

Ragged Arrays



A standard two-dimensional array is a rectangle.

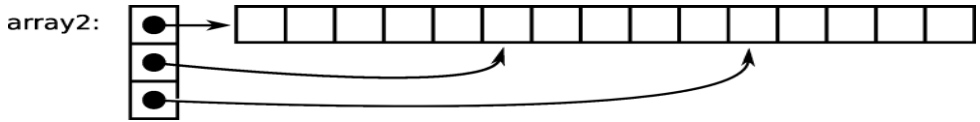


With Variant arrays, you need not waste space.

Ragged Array

```
dimensions:
  station = 193;
  obs = 876;
variables:
  float data(obs);
    :coordinates = "lat lon time";
  float count(station);
    :sample_dimension = "obs";
  float lat(station);
  float lon(station);
  int time(obs);
```

Contiguous ragged array representation of time series



Forecast Models, two times Whats wrong?

```
dimensions:
  reference = 30;
  forecast = 24;
variables:
  float data(reference, forecast, y, x);
    :coordinates = “lat lon time”;
 float reference(reference);
    :units = “days since 01-01-2015”
 float forecast(forecast);
    :units = “hours since 01-01-2015”
```

Correct

```
dimensions:
  reference = 30;
  forecast = 4;
variables:
  float data(reference, forecast, y, x);
    :coordinates = “lat lon time”;
 float reference(reference);
    :units = “days since 01-01-2015”
 float forecast(reference, forecast);
    :units = “hours since 01-01-2015”
```

Coordinate System Rules

1. Coordinate = Coordinate Variable or Auxiliary Coordinate Variable
2. Coordinate System = set of Coordinates
3. Define a Coordinate System for a Variable

   ```
   data:coordinates = "lat lon time altitude";
   ```

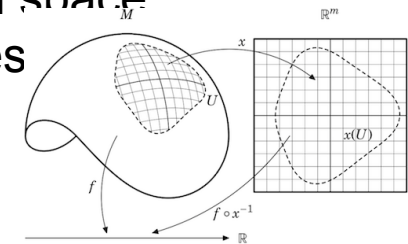
4. The dimensions of a Coordinate $\subset$ dimensions of the Variable*

   ** except for ragged arrays*

5. The number of Coordinates = dimensionality of the embedding space*

   ** only for georeferencing coordinates, eliminating duplicates*

6. The number of Dimensions = dimensionality of the embedded space
7. The number of Dimensions = number of independent variables



Variables as Sampled Fields

- *Field* is a function on the vector space of real numbers: $F : R^n \rightarrow R^m$
- *Variable* represents a sampled Field, a multidimensional array containing the values of the Field at some set of points: float data(time, level, lat, lon)
- The sampling is specified by *dimension*s = $D_1 \otimes D_2 \otimes D_3 \ldots = (0..n_1\text{-}1) \otimes (0..n_2\text{-}1) \otimes ..$, the domain of the variable in Index Space: (time, level, lat, lon) = $\prod D_i = D^n$
- A Variable is then one of the scalar component functions of F: $V:D^n \rightarrow R$
- A *Coordinate System* CS locates the values of the samples, typically on the earth in real space/time: $CS:D^n \rightarrow R^n$
- A *Coordinate Axis* is one of the components of the vector function CS: CoordAxis (time, level, lat, lon) ➔ R, common case is one dimensional: lat(lat)
- A CS needs to be invertible to find the array indices from the coordinates: CS^{-1}: $R^n \rightarrow D^n$ in order to find Field's value: $F = V \circ CS^{-1}:R^n \rightarrow R^m$
- The Variable and CS must have same domain, that is, share dimensions